

< The Gaming Room >

# **CS 230 Project Software Design Template**

# **Draw It or Lose It**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <11/17/22> | <Joshua Perez> | <Original submission of project 1> |
| 1.1 | <11/28/22> | <Joshua Perez> | <Original submission of project 2> |
| 1.2 | <12/11/22> | <Joshua Perez> | <Original submission of project 3> |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

The gaming room wants to make a web-based game that will inherently serve to operate on multiple platforms. The game that they want to develop is based on their currently existing game, “Draw it or Lose it”, which is available only on the android platform. The application will render images from a large library of stock drawings. Every game will consist of four rounds, each lasting one minute, and will have one or more teams able to join. The drawings will be rendered at a consistent rate, taking 30 seconds to be fully revealed. If failed, the other team gets the opportunity to answer with a 15 second time limit. Each team will have multiple players assigned to it. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name. Only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

The design constraints for developing a game application in a web based distributed environment fall within the limitations of server-based configuration as everything must be hosted and must have scalability considered, this needs to be kept in mind due to the need to store a library of stock drawings and have unique identifiers for each instance of a game, team, or player. There should be a standard of uniformity present across all versions of the game, therefore the constraint will be in building a web version of the game that runs smoothly on different devices using a web browser. In other words, making sure that the web application will work regardless of the underlying platform in which the web application is being accessed. This will involve porting over the working android code available into a web application.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_8h2ehzxfam4o)

The entity clasts created in this most recent iteration is the parent / super class of the Game, Team, & Player classes. These classes inherit the attributes of the super class, Entity. In other words, these classes will all include the attributes “id” and “name”. The Game Service class references Game. Game references the Teams class, the Teams class references the Player. This means that Game Service can contain multiple games, and each game can then have its own teams, and each team can have its own players. This is a factor of aggregation. Encapsulation can be seen between the Entity Class and the Game, Team, and Player classes. An example of a singleton in this would be the GameService() located under the GameService Class.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The Mac OS does have a built-in web server in the form of “Apache”. Due to the innate nature of the machine being used to host, it will likely be more expensive than the alternatives. This comes with the upside of having secure hosting options with high uptime. While this seems to be the most expensive option, in some ways it is probably the best option for hosting a reliable game web application. | Linux OS has the advantage of needing the least number of resources to function on the low end. It also opened sourced and hardly gets updated, so the uptime is quite good. The web host can implement their own security as well. Linux is likely the most cost-effective option of the bunch and can be quite competitive with Mac’s OS X servers. | Windows is an option that can be quite variable in quality, without high end gear you could face quite a large amount of strife when it comes to running it. It is probably the least effective of the solutions given here but can be used for specific purposes like the Microsoft-proprietary web applications that require it. Grid based hosting can eliminate the downtime that may occur. | Mobiles devices are not many people’s platform of choice when it comes to servers as it is the least reliable and least powerful of the choices. The main advantage comes from the fact that they are incredibly low in cost. |
| **Client Side** | The cost for users on Mac clients would likely be higher due to the hardware required and the fact that the OS is not open sourced. The time sink would depend entirely on the user, but these will be similar to windows-based installations which are usually quick and easy. The expertise required is minimal, but there will be some need for familiarity with the platform. | The cost for users on Linux clients will be low, if any, as the system is open sourced. Time and expertise might be required more so on this platform than other platforms as the average user does not have much familiarity to it. | The cost for Window’s users will be comparable to Mac as it is not open sourced, but it will usually demand less cost from the client side than MAC. The time and expertise required is also comparable to other systems and most may find this easier due to the popularity of the platform. | The cost and experience required on a mobile platform is not much of an issue and is usually the easiest of the bunch for the average person to work with. Time on the other hand will be needed as there are a lot of different mobile devices on the market to work with. |
| **Development Tools** | The relevant programming language when working with Mac is SWIFT, as it is the most common language for developing on the platform. Among the most popular IDEs for SWIFT there are Atom and Visual Studio code which are my personal favorite. | When working with Linux the relevant programming languages are many, however it is primarily used with Java and other popular languages like C++. The most commonly used IDEs are Eclipse and ATOM. | When working with windows the relevant programming languages are HTML, C#, and Java Script. The most commonly used IDEs are Eclipse and Visual Studio. | For the popular iPhone, the tools used are very similar to those of MAC and will likely use the SWIFT language. For Android devices, the majority of apps are written in Java and Kotlin. IDEs used for this are the same as most of the other platforms but can also use native Android IDEs such as JavaIDEdroid or Java Editor. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.

If I were choosing to recommend an appropriate operating platform for the servers of The Gaming Room to expand through, I would likely suggest that they use Linux servers. Not only would they be massively less expensive than the competitors, but they will have sustainable uptime and good performance. Linux’s open-source operating system allows for many solutions to the problem of server hosting, for The Gaming Room application we’d probably stick to a “imagine the game goes viral” mentality and host a server that can scale higher such as Red Hat® Enterprise or Ubuntu. According to CompTIA, “Linux scalability is inherent to Linux and comes with every GNU Linux system by default.” This means the package manner that Linux has will make it easier to upscale the created web application to other computing environments. Linux also has support for hyperthreading and is able to incorporate live changes to certain aspects without having to reboot.

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.

The Linux architecture is largely composed of elements such as the Kernel, System library, Hardware layer, system, and shell functions. The kernel is responsible for the primary duty of the Linux OS and is in charge of creating an appropriate abstraction for concealing trivial hardware or application strategies. The system library is a set of library functions that may be specified as these functions. The system utility program performs individual and specific jobs. The hardware layer is made up of the system peripherals such as CPU, HDD, and Ram. The shell of the OS is represented by two different categories, graphical shells and command-line shells. The graphical shell allows for graphical user interfaces, while the command line shell enables command line interfaces. TPM and disk encryption will protect storage devices and RAID systems will provide reliability, performance, and capacity.

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.

An appropriate storage management system that can be used with Draw IT or Lose IT, is likely to be a cloud storage one. Keeping the data away from user’s phones and purely on server-side storage means that we can lower the barrier to entry for users. This can also be used to load the images being used each round on a game-by-game basis instead of having it always loaded in the system. laaS aka Linux as a service is not only cheap and quick to set up, but also as non-constraining as a service can be. laaS provides high level API’s capable of deference of low-level details in the underlying network infrastructure. This will help with the managing of physical computing resources, location, data portioning, scaling, security, backup etc. We can use memory caching to temporarily store data on the RAM, keeping short term things like game instances and visuals only temporarily before unloading them and using them for another game. Garbage collection will provide automatic memory management and stack allocation will automatically deallocate and flush out the data.

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.

Linux has the potential to be very fast and powerful and, with the use of laaS for memory management, it can provide storage backup & recovery, simplifying the system overall. Therefore, this is a solution that can provide results for storage and memory management. Since the infrastructure is cloud based, it can scale more easily to the number of users on the system that are utilizing resources. This will also allow for data to be temporarily loaded and unloaded on demand. When it comes to Draw It or Lose it, laaS will release anything no longer in use in order to save memory and use what is available more efficiently. By using a retain count, or reference count to assign a value to an object, that object will only exist until the code reaches the end of a code block unless told to do otherwise in some form. We can use a similar method to represent the images in Draw It or Lose it, only loading them into cached memory when they become essential for the program, keeping them in reserve on the HDD until they are needed.

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).

Luckily this is something that can be solved via relatively simple methods since the solution we will be using is the laaS. When I say relatively simple, it is a bit of an understatement, but it is doable enough that I’ve been able to do a mini test of it on my own with only a little bit of knowledge and documentation. Everything will be hosted on an open-source Linux server which can be made to be adaptable to any system in which it is used. It is also very cost effective and good for scalability. A load balancer can be used for the optimization of and response times and prevent overloading. The use of a RAID system can be used to provide redundancy for the system. RAID 10 can be used to provide security by mirroring all the data on a 2nd drive, using striping to speed up data transfers.

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.

This is one of the areas where Linux OS can succeed highly when it comes to server-based security. It is a platform known for its security prowess and it is also under strict regulation and has strict security requirements. The provider of the cloud service will likely also have a team of experts dedicated to making sure the application is secure at all times. laaS also provides encryption capabilities and a system can be made to encrypt data at all times. Ports will be properly configured, multi-factor authentication will be activated, and storage access will be closed. These measures in conjunction with good programming practices, root certificates from trusted certificate authorities, and proper front end encryption will ensure that everything is secured from beginning to end. Identifying a threat model will help programmers inspect what’s been implemented. Input validation on client side and server side can also be added to ensure that even the most basic of basics is covered. Transport layer security or TLS protocol can be used to ensure confidentiality in the data that is be transferred. When it comes to encryption algorithms, I would stick with SHA 256 or SHA 3. A good security information event management system can be used to effectively manage security alerts and keep user information safe. Another option would be using security as a service aka SaaS as they would cost money, but likely be far more effective than anything they could manage on their own.